Jackknife: A Reliable Recognizer with Few Samples and Many Modalities

Eugene M. Tarnata II
etaranta@gmail.com

Amirreza Samiei
samiei.amirreza@knights.ucf.edu

Mehran Maghoumi
mehran@cs.ucf.edu

Pooya Khaloo
pooya@cs.ucf.edu

Corey R. Pittman
cpittman@knights.ucf.edu

Joseph J. LaViola Jr.
jjl@cs.ucf.edu

University of Central Florida
Orlando, FL 32816, USA

ABSTRACT

Despite decades of research, there is yet no general rapid prototyping recognizer for dynamic gestures that can be trained with few samples, work with continuous data, and achieve high accuracy that is also modality-agnostic. To begin to solve this problem, we describe a small suite of accessible techniques that we collectively refer to as the Jackknife gesture recognizer. Our dynamic time warping based approach for both segmented and continuous data is designed to be a robust, go-to method for gesture recognition across a variety of modalities using only limited training samples. We evaluate pen and touch, Wii Remote, Kinect, Leap Motion, and sound-sensed gesture datasets as well as conduct tests with continuous data. Across all scenarios we show that our approach is able to achieve high accuracy, suggesting that Jackknife is a capable recognizer and good first choice for many endeavors.

ACM Classification Keywords

H.5.2 Information interfaces and presentation: User interfaces, input devices and strategies; I.5.5 Pattern recognition: Implementation, interactive systems

Author Keywords

Gesture Recognition; Dynamic Time Warping; Rapid Prototyping; Gesture Customization; User Evaluation.

INTRODUCTION

Although gesture recognition has been an active area of research for some time, it has become especially popular in the last decade with the introduction of several low-cost interactive devices, such as the Wii Remote, Kinect, and Leap Motion. To support gesture recognition on these devices, a flurry of techniques have been proposed and evaluated, many of which rely on domain specific knowledge to achieve competitive accuracy (above 90%). And unsurprisingly, support vector machines, hidden Markov models, conditional random fields, decision trees and random forests, as well as variants of these machine learning approaches are ubiquitous [16]. However, these popular approaches are also inappropriate for rapid prototyping or gesture customization given that either copious training data or advanced knowledge of machine learning are typically required.

For the rapid prototyping community, gesture recognition must be easily accessible[1] for development on new platforms, where best approaches have not been identified, or where libraries and toolkits may not yet be available and consumers are unfamiliar with machine learning. The $-family of recognizers [1, 2, 17, 35, 38] has been effective at addressing this issue for pen and touch, though attempts to generalize these techniques to higher dimensionalities have been less successful. For gesture customization, recognizers must work well with as little as one training sample per gesture class, as users may be unwilling to provide a plethora of samples for a variety of classes. Even when we relax these constraints, most research still only focuses on one domain or input device, *e.g.*, Kinect, Leap Motion, or Wii Remote for full body or hand gestures, and rarely does research consider general methods that might easily be adopted to any modality. Another common issue is that many researchers evaluate their methods using segmented data without addressing continuous data stream related issues. As a result, it remains unclear and confusing what might be a good starting point when one desires to incorporate gesture recognition into their work.

We begin to address these issues with Jackknife, a general recognizer for dynamic gestures that, much like a multitool, is a collection of techniques designed to handle various scenarios. Our recognizer is designed to be modality-agnostic, so that little domain specific knowledge is required, and competitive accuracy can be achieved with minimum training data. At the heart of Jackknife is dynamic time warping (DTW), an elastic dissimilarity measure suitable for handling large gesticulation variability. As DTW has repeatedly been shown to be a high quality recognizer in time series research, especially for nearest neighbor (NN) pattern matching, it is surprising to

[1]By easily accessible we mean self contained, easy to debug, and can be implemented without too much effort.

CHI 2017, May 06-11, 2017, Denver, CO, USA
Copyright © 2017 ACM. ISBN 978-1-4503-4655-9/17/05$15.00
DOI: http://dx.doi.org/10.1145/3025453.3026002

us that this technique is not already the de facto go-to gesture recognition method. Still, since DTW itself was not designed explicitly for gesture recognition, we introduce several extensions that improve its base accuracy under various conditions.

While the main contribution of this paper is a general, device-agnostic dynamic gesture recognizer, several additional contributions are made to achieve this objective, which are that we promote and raise awareness that DTW is a powerful 1-NN gesture recognizer useful for rapid prototyping and custom gestures; demonstrate that the inner product of gesture path direction vectors is often a better local cost function for DTW as compared to the pervasive squared Euclidean distance, which to the best of our knowledge has not been explicitly evaluated; present new correction factors that modify the DTW score and help disambiguate certain gestures as well as further separate negative samples from positive sample; and introduce a new method to select a per class rejection threshold for gesture spotting in a continuous stream and evaluate this method with a new dataset collected from 40 participants. Finally, reference source code and our dataset is available at http://www.eecs.ucf.edu/isuelab/research/jackknife.

The rest of the paper is organized as follows. First, we present related works where we focus on what motivates Jackknife, after which we describe our recognizer in detail. With an understanding of the core methods in hand, we subsequently evaluate the various aspects of Jackknife by analyzing visualizations of score distributions, and by performing recognition tests with segmented and continuous data. Finally, we discuss our findings.

RELATED WORK

Since space precludes a comprehensive discussion of all gesture recognition work, we direct the reader to few surveys [12, 21, 25, 30, 31]. For a taste, Mitra and Acharya [21] discuss different gesture recognition methods such as hidden Markov Models (HMM), particle filtering and condensation, finite-state machines (FSM), and neural networks. As hand-based gesture recognition is a popular topic, Suarez and Murphy [31] look through recognition methods using depth images. These methods include common and customized approaches like Du's [8] classifier that counts the number of convex points in a hand silhouette for classification over a small set of static poses. Ibraheem and Khan [12] also discuss HMMs, neural networks, and histogram based feature and fuzzy clustering algorithm methods. Sarkar *et al.* [30] similarly report on both 2D and 3D methods for hand gesture recognition. It is worth noting that computer vision techniques such as those just mentioned are common in the gesture (and action) recognition literature; however, in this work we assume that gestures are a time series of trajectories through a multidimensional space.

A common theme of these works is that a large amount of training data is needed to train the recognizers. Another issue is that many approaches also require domain-specific knowledge to extract features useful for discriminating gesture classes, which in itself can be a very challenging problem to solve. Two clear advantages of DTW-based recognizers are that competitive accuracy is still possible with limited data and complex feature extraction is unnecessary.

Why DTW and Not Some Other Measure?

Gestures are naturally represented as time series, and it has long been known that DTW is an excellent measure for 1-nearest nearest (1-NN) pattern recognition in times series classification problems. As an example of its power, Giusti and Batista [10] compared 48 dissimilarity measures over 42 time series datasets and found that DTW and CIDDTW [4] (which is incorporated in this work) are among the top performing measures on average. More recently, Bagnall *et al.* [3] conducted an extensive evaluation of 18 recently proposed state-of-the-art classifiers over 85 datasets and found that many of the approaches do not actually outperform 1-NN DTW or Rotation Forest, and they also remark that DTW is a good base to compare against new work.

DTW has also already been used quite successfully in gesture recognition. Celebi *et al.* [6] implemented weighted DTW for Kinect to recognize 8 gestures with 28 samples per gesture. They weighted each skeleton joint differently by optimizing a discriminant ratio. Wu *et al.* [39] utilized DTW for user identification and authentication on Kinect. Their dataset consisted of 8 unique gestures, each repeated 5 times, collected from 20 individuals. Bodiroza *et al.* [5] used DTW on Kinect data to recognize 6 hand gestures performed 20 times by a single subject. Vikram *et al.* [36] developed a DTW-based recognizer for the Leap Motion sensor to recognize hand writing in the air. Their dataset consisted of both uppercase and lowercase alphabetical letters, and each letter was repeated 5 times by 100 participants. Not only do we see again that a large number of training samples are used, but also these works mostly utilize Euclidean distance as the local cost function for DTW. In our approach, we emphasize the inner product of gesture path direction vectors and show that this local cost function usually achieves higher accuracy.

What About Speed... I Hear It's Slow?

DTW is infamous for its sluggishness, which we believe is an unfair assessment. In its classic form, DTW is quadratic; however, Rakthanmanon *et al.* [23] demonstrated that DTW can be optimized to achieve search times faster than typical linear time Euclidean distance search algorithms. Further, even without optimizations, Vatavu [34] found that low sampling rates achieve high accuracy for Euclidean distance, angular, and DTW based 2D gesture recognizers. In this work, we resample gestures to $n = 16$ points; and by using a Sakoe-Chiba Band [29] (to control pathological warping) that constrains DTW's search window to $r = 2$, complexity drops to $\mathcal{O}(rn)$. It is also useful to compare DTW with \$P [35], a $\mathcal{O}(n^{2.5})$ recognizer: specially, we note that \$P is popular and in common use, suggesting that the algorithmic complexity of DTW should not be an issue for many applications.

How Can We Select a Rejection Criteria?

Liu and Chua [19] summarize three common approaches for rejecting negative samples: build a set of garbage models from a set of explicitly defined unwanted patterns, learn a cut off based on the distribution of scores between classes, or generate mixture models from positive samples to form a *universal background model* (UBM). These are general methods used across various fields and in particular the UBM approach stems from speaker recognition [26]. In all of these cases, there is a sufficient amount of training data to construct representative probability distributions, which is a not a luxury we have with only one or two training samples per gesture class. Our approach is most like UBM, where negative samples are synthetically generated from positive samples, although we do not generate an explicit background model. Instead we use the distribution of scored synthetic negative and positive samples to help select a rejection threshold. To increase the positive sample distribution, we synthetically generate new samples using a recently introduced technique, gesture path stochastic resampling [32].

JACKKNIFE

Jackknife is a multitool for gesture recognition equipped with the following functionalities: dynamic time warping using the inner product of gesture path direction vectors as the local cost function and squared Euclidean distance as a back up, correction factors that inflate the scores of dissimilar gestures, and a synthetic gesture generator for learning a rejection criteria. In practice, one will only need to implement the components required for their specific application.

In this work, we treat gestures as time series, an ordered set of points:

$$P = (p_i | i = 1..n), \tag{1}$$

where n is the number of points in the series and each $p_i \in \mathbb{R}^m$. Typical values of m for various modalities include $m = 2$ for pen or touch, $m = 21 \times 3$ for Kinect 2.0, and $m = 21 \times 3$ for Leap Motion. A gesture can also be represented as a set of unit length direction vectors through m-space, which we refer to as the *gesture path direction vectors*:

$$\vec{P} = \left(\vec{p}_i = \frac{p_{i+1} - p_i}{|\ p_{i+1} - p_i\ |} \ \middle|\ i = 1..n-1 \right). \tag{2}$$

Further, we denote a query sequence as Q and a template sequence as T, where a template is a time series representation of a specific gesture class. Jackknife employees 1-nearest neighbor classification, where a query is compared against all templates stored in a database, and the template whose measure against the query is least dissimilar is said to match the gesture class of the query. Formally, given a set of templates $\mathbb{T}$ and a query Q, the query's class is the gesture class of $T_i \in \mathbb{T}$ that minimizes their dissimilarity:

$$match = \underset{T_i \in \mathbb{T}}{\operatorname{argmin}} \prod_{j=1}^{F} f_j\left(T_i, Q\right), \tag{3}$$

where f_1 is the DTW measure of T and Q, and $f_{2 \leq i \leq F}$ are correction factors. This approach is based on the complexity-invariant distance (CID) measure [4]. CID inflates a base distance measure, such as Euclidean distance or DTW, with a correction factor (CF) that is a measure of the complexity difference between a template and query. CF ranges from one to positive infinity such that time series similar in "complexity" score near one and the base distance measure remains relatively unchanged. Otherwise, the score is inflated. Our interpretation of CID is that a good CF is able to capture information about the dissimilarity of two time series for which the base distance measure is unable, though the CF measure does not necessarily need to relate to notions of complexity.

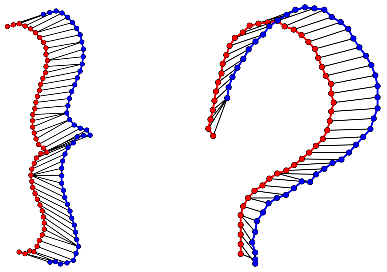
Figure 1: Visualization of the 2D alignment found by DTW between two right curly braces (left) and two unistroke question marks (right), from the $1-GDS dataset [38].

Dynamic Time Warping

Dynamic time warping (DTW) is a dissimilarity measure that allows local warping of time series in order to find the best alignment. Given two time series T and Q of length n and m, we construct an n-by-m cost matrix. Each element (i, j) in the matrix stores the minimum cumulative distance between the subsequences $t_1, ..., t_i$ and $q_1, ..., q_j$. This cumulative distance can be found efficiently with dynamic programming by solving the following recurrence:

$$\gamma(i,j) = \ d(t_i, q_j) + min \begin{cases} \gamma(i-1, j-1) \\ \gamma(i-1, j) \\ \gamma(i, j-1), \end{cases} \tag{4}$$

where $d(t_i, q_j)$ is a local cost function, discussed shortly. In other words, during the evaluation of element (i, j), the cumulative measure of three neighbors are considered, the minimum is taken, and the local cost of matching t_i to q_j is added to the minimum, which then becomes the minimum cumulative distance for the subsequences under evaluation. Once the matrix is fully evaluated, element (n, m) is the DTW score for T and Q. The path through the matrix that defines the minimum cumulative distance between the sequences is the optimal warping path, which is a set of alignments between T and Q. A visualization of the warping path between two different 2D gestures is shown in Figure 1.

Local Cost Function

The local cost function $d(t_i, q_j)$ in Equation 4 is most frequently the squared (or standard) Euclidean distance over z-score normalized sequences[2]: $d(t_i, q_j) = (t_i - q_j)^2$. Though, we find in gesture recognition that this cost function is not always the best option. An alternative that has received less attention is the inner product of a feature vector measure [20], which can also be applied to gestures. Instead of extracting feature vectors from a time series, we utilize the gesture path direction vectors (see Equation 2). Since the inner product is a similarity measure in $[-1, 1]$ for unit vectors, we convert this to a dissimilarly as follows:

$$d(\vec{t}_i, \vec{q}_j) = 1 - < \vec{t}_i, \vec{q}_j > \tag{5}$$

[2]Each sequence is z-score normalized independently.

where $1 \leq i < n$. This approach is inspired by the 2D Penny Pincher [33] gesture recognizer that also uses the inner product of direction vectors, an approach that proved to be empirically faster than alternative unistroke recognizers while remaining competitive in accuracy. When used as the local cost function, we will later demonstrate that the inner product measure (IP) is often superior to the squared Euclidean distance measure (ED) in gesture recognition problems.

Warping Window Constraint

With classic DTW, pathological warping can occur when a small part of one subsequence is inappropriately mapped to a large portion of another [24]. To prevent this issue, one can constrain the amount of warping allowed. A popular approach is the Sakoe-Chiba Band [29], which limits the maximum distance r the warping path can stray from the diagonal, where $| i - j | \leq r$. An additional benefit immediately apparent is that constraining the warping path can significantly speedup DTW as less of the cost matrix is evaluated. The optimal warping window varies per problem [24], but a very common constraint is 10% of the time series length, which also yields very good results in our testing.

Lower Bounding DTW

The cost of performing DTW is not much of a concern when working with segmented data at a low resampling rate as well as with a small number of gesture classes and training samples. However, when working with a continuous data stream where DTW evaluations are frequent and observational latencies are problematic, it can be useful to prune templates that will obviously not match a query. An efficient lower bound (LB) score, where $LB(T,Q) \leq DTW(T,Q)$, can be used to avoid a full evaluation in two cases: when the determined lower bound is worse than a predetermined rejection threshold or when a closer match has already been found. One such lower bound for DTW using ED is LB_{Keogh} [13], which envelopes a time series T with an upper (U) and lower (L) band based on a window constraint r:

$$U = \left(u_i = \max_{i-r \leq i \leq i+r} t_i \right) \quad L = \left(l_i = \min_{i-r \leq i \leq i+r} t_i \right). \tag{6}$$

Note that $\forall_i\ L_i \leq t_i \leq U_i$. To lower bound a query Q against a template T for DTW using ED, we define:

$$LB_{Keogh}(T,Q) = \sum_{i=1}^{n} \begin{cases} (q_i - U_i)^2 & \text{if } q_i > U_i \\ (q_i - L_i)^2 & \text{if } q_i < L_i \\ 0 & \text{otherwise} \end{cases}, \tag{7}$$

which is visualized in Figure 2.

A lower bound has also been derived for the inner product of posteriorgrams [40] based on the local cost function $d(t_i, q_j) = -log \langle t_i, q_j \rangle$. One issue with this inner product lower bound is that each component in a posteriorgram is non-negative, which is unlike gesture path direction vectors whose components can take on negative values, and hence a different lower bound is required. Our inner product lower bound is similar to LB$_{Keogh}$, where the upper and lower bands are calculated in the same way. However, the summation instead is based on the IP distance. If a component in the query vector is positive, then its similarity will be maximized when paired with the upper band

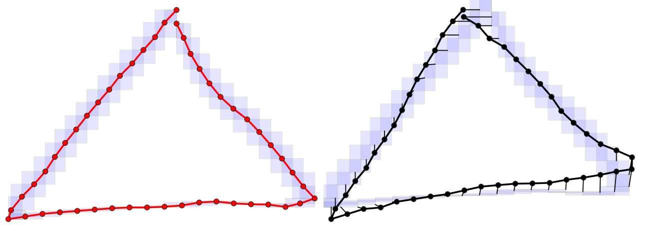

Figure 2: Visualization of a LB$_{Keogh}$ lower bound in 2D for the triangle gesture from $1-GDS [38]. On the left, the combined upper and lower bands of a template form a light blue bounding box per point. On the right, the LB$_{Keogh}$ lower bound is calculated for a query as the sum of the minimum squared Euclidean distance from each point in Q to the corresponding point boundary from T (shown as thin black lines).

component (if both signs are positive), but the dissimilarity is also minimized when paired with the upper band (if the signs are opposite). For the same reasons, if a component in a query vector is negative, pairing with the lower band will always produce the best possible result, which leads to the following IP lower bound:

$$\begin{aligned} LB_{IP}(T,Q) &= \sum_{i=1}^{n} 1 - min\left[1, \max\left(-1, < lb_i, q_i >\right)\right] \\ lb_i^j &= \begin{cases} U_i^j & \text{if } q_i^j j \geq 0 \\ L_i^j & \text{otherwise} \end{cases}, \end{aligned} \tag{8}$$

where j indexes a vector's components ($1 \leq j \leq m$) and the template and query sequence are assumed to be unit length. This lower bound can be proved using the same techniques as in [13, 40].

Correction Factors

After the DTW score is calculated, correction factors are applied that inflate this score based on additional dissimilarities not as easily perceived by only time warping measures, a method that we adopted from CIDDTW [4]. Our correction factors, however, use the inverse inner product of normalized feature vectors:

$$f_i(T,Q) = \frac{1}{< g_i(T),\ g_i(Q) >}, \tag{9}$$

where $2 \leq i \leq F$ per Equation 3 and each g_i transforms the time series into a *normalized* vector whose dimensionality is greater than one (otherwise its normalization would simply result in a scalar equal to one). The vector must also comprise only non-zero components so that the denominator of Equation 9 falls in $(0,1]$ and the domain of f_i becomes $[1,\infty)$. Intuitively, times series that are similar should score near one so that DTW score inflation is minimized.

We now describe two correction factor transformations that are designed specifically for gestures and are inspired indirectly by Rubine's features for pen based gesture recognition [28]. In both cases, we focus on between-component information that is scale and position invariant. These latter attributes are important to ensure that differences between users do not appear as dissimilarities in their gesticulations. First, the component-wise absolute distance traversed by gesture P is given by:

$$g_{abs} = \sum_{i=1}^{n-1} |\vec{p}_i| . \tag{10}$$

Once g_{abs} is normalized, the components of this vector yield a relative measure of the total distance traveled by each component — this correction factor helps to ensure that the contributions to the gesture path for each component of two different samples are similar in measure.

Our second feature is based on the bounding box extents of the concatenated *unnormalized* direction vectors in $\mathbb{R}^m$ space:

$$\begin{aligned} g_{bb} &= bb_{max} - bb_{min}, \\ bb_{max} &= \left(bb_{max_j} = \max_{1 \le i < n} \sum_{i=1}^{n-1} p_{i+1}^j - p_i^j \right), \\ bb_{min} &= \left(bb_{min_j} = \min_{1 \le i < n} \sum_{i=1}^{n-1} p_{i+1}^j - p_i^j \right). \end{aligned} \tag{11}$$

Once normalized, we have the relative distances spanned by each component. We found this bounding box correction factor to be useful in compensating for gestures that are mostly similar, except in span. Consider two gestures: *climb ladder*, which is a hand over hand action, and *balance*, which is a seesaw-like action with arms extended [9]. In both cases, the hands, wrists, and elbows oscillate vertically. The only major difference is that one's arms are extended outward to perform the balance gesture. This extension, which occurs at the beginning of the action, represents only a small duration of the entire gesture and may be difficult to detect with either ED or IP, depending exactly on how a participant performs either gesture. However, the bounding box (span) of each gesture is very different, which is why using the bounding box as a correction factor can improve recognition accuracy.

Rejection

How one can determine an appropriate per template rejection threshold from only a minimum amount of training data remains a difficult problem. With sufficient amounts of training data, a recognizer can estimate within class score probability densities and select thresholds sufficiently low enough to prevent type II (false negative) errors. One can also use negative samples that are non-gesture sequences to help control type I (false positive) errors by ensuring that a rejection threshold is sufficiently high enough to prevent false positives. With access to both positive and negative samples, one can instead select a threshold that minimizes both error types, which is the strategy we adopt. However, in this work, we assume that only a minimum number of positive training samples are given, as little as one or two per gesture class. This limitation implies we need to somehow synthesize both negative and positive samples.

To create negative samples, positive samples are spliced together to create semi-nonsense, noise-like sequences. We favor this approach because we desire that negative samples have parts of real gestures embedded within to ensure that Jackknife can reject sequences that partially resemble but are not actually real gestures. To generate synthetic negative samples, we randomly sample k training samples and from each sample we randomly sample $(n-1)/k$ sequential direction vectors. These direction vectors are concatenated together to form a negative sample. We then compare each template with the negative sample using DTW and save the results per template. This process is repeated a number of times, after which the scores are z-score normalized per template.

The generation of synthetic positive samples requires a different approach. Gesture path stochastic resampling (GPSR) [32] is a new synthetic data generation developed specifically for 2D gestures and rapid prototyping. A gesture path is non-uniformly resampled to $n+x$ points, after which the distance between subsequent points is normalized to unit length, and finally x random points are removed. GPSR was shown to produce realistic results for pen and touch gestures; however, for our use, we do not require realism. Rather, we only need to be able to create a distribution of samples that when evaluated with DTW generates a score distribution similar to the true distribution, and we found GPSR works well for this purpose. For additional information on implemented details and pseudocode, we refer the reader to [32]. In Jackknife, we use GPSR to create synthetic positive samples that are scored with DTW against their seed samples. These within class scores are then z-score normalized using the mean and standard deviations generated from the negative samples evaluation above.

Now we are able to determine a rejection threshold. With the distribution of positive and negatives samples (all of which have been z-score normalized), a standard deviation λ is selected that minimizes the aggregate false and negative positive count. Since we trained with a noise-like pattern, our goal is to be able to reject idle-like motion[3]. The per template rejection threshold δ is then:

$$\delta_i = \mu_i - \lambda \sigma_i, \tag{12}$$

where μ_i is the mean of the negative sample scores relative to each template i and σ is its standard deviation. As we will show in an evaluation, this approach appears to select thresholds near the optimum.

EXAMINATION OF THE DISTRIBUTIONS

To examine the distribution of scores generated by Jackknife for a specific dataset, we do the following: For a given subject, a recognizer is trained with one random sample per gesture class. All z-score normalized synthetic scores generated as part of the training process are saved for analysis. An additional, but unique test sample per gesture class is subsequently evaluated, and the within class DTW score of each test sample, before and after correct factor inflation, are similarly z-score normalized and saved. Last, new synthetic negative samples are again generated, scored, and saved. This process is repeated 10 times per subject and all results are combined into a single set of distributions.

Results are shown in Figure 3, and we note a number of observations. First, the within class samples distribution is well

[3]To increase tolerance to higher energy patterns, the minimization can non-uniformly weight the importance of true and false positives.

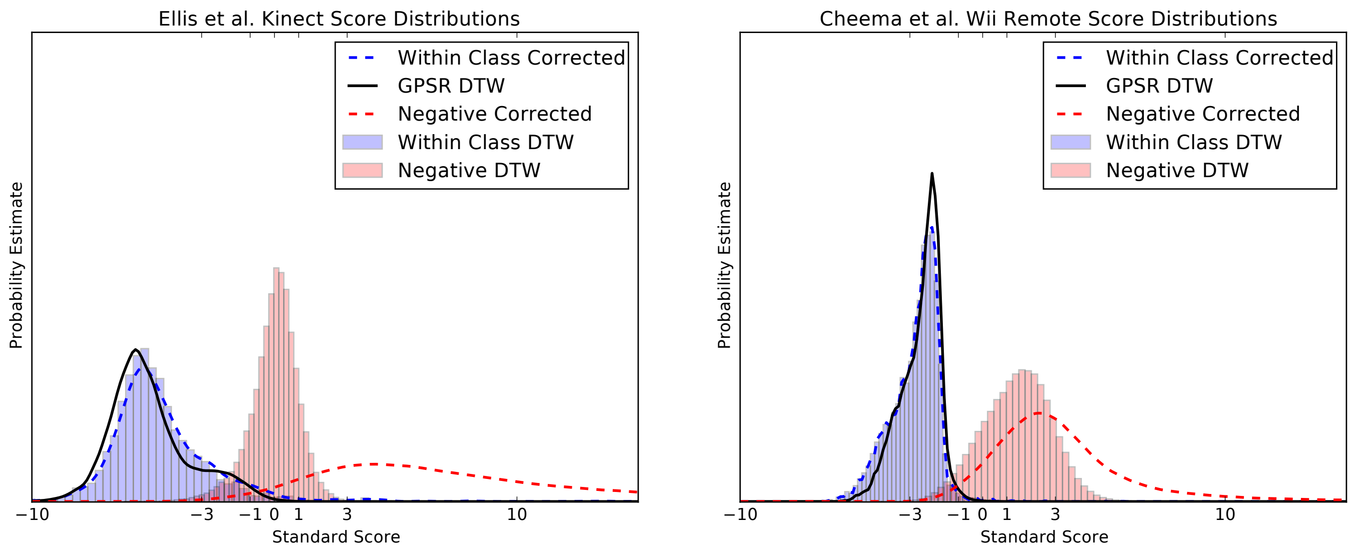

Figure 3: Normalized distributions of within class and negative samples before and after correction factors are applied, as well as the synthetic positive sample distribution for Ellis *et al.*'s [9] and Cheema *et al.*'s [7] datasets.

separated from the negative samples, having only a small amount of overlap, which we perceive as a desirable property for a noise pattern; a poor noise distribution would be far away from the within class samples and result in unreliable rejections thresholds (no false positives or false negatives). Second, the within class corrected DTW scores are remarkably close to the true distribution of the uncorrected DTW scores, whereas the distribution of negative samples are shifted right, away from the within class distribution. This observation suggests that the correction factors are doing their job, although further analysis in the next section will quantitatively confirm this. Finally, the positive samples generated using GPSR form a score distribution that is quite near the true distribution. This does not imply that the synthetic samples are realistic, but it does help build confidence that GPSR can be used to help find a reasonable rejection threshold when used in combination with synthetic negative samples.

EVALUATION OF SEGMENTED DATA

In order to evaluate Jackknife, we consider a number of publicly available gesture datasets that span a range of input modalities. Where appropriate, we compare our recognizer against alternative methods. In general, we start with low dimensional data and work our way into higher dimensions. We perform repeated measures full factorial ANOVA analysis to understand if there are statistically significant differences between the recognizer variants evaluated. However, because recognition error rates are often quite low, accuracy measures are often non-normally distributed and may also violate the homogeneity of variance assumption, which is why we utilize the Aligned Rank Transform method for ANOVA analysis [37]. In the ANOVA results tables, the measure factor is either euclidean distance (ED) or inner product (IP), and the correction factors (CF) are either disabled (False) or enabled (True). Further, $T = x$ specifies that x samples (templates) per gesture class are used for training.

Pen and Touch

Table 1 presents results for Jackknife variants on the $1-GDS pen and touch dataset [38]. This dataset contains 4800 samples of 16 pen gestures collected from 10 participants. Our results were obtained using a writer-independent protocol. The reason for focusing on writer-independence is that state-of-the-art recognizers already achieve near perfect recognition rates in writer-dependent tests on this dataset. Given a training participant, our writer-independent protocol randomly selects T samples per gesture class from that participant for training. Next, one sample per each gesture class is randomly selected from the pool of samples from all other participants, which is used as the test set. This sampling process is repeated 500 times, and all results for the training participant are combined into a single average accuracy value for that individual. This process is repeated for each participant, so that each is used to supply training data.

We observe that using inner products (IP) with gesture path direction vectors yields higher accuracy with significance, as does increasing the template count. Correction factors were not a significant factor; however we cannot say how the correction factors affect negative samples with this test. In Table 2 we compare the best Jackknife variant to $1 [38] and $P [35], two popular rapid prototyping $-family recognizers. The main observation is that Jackknife outperforms the alternative recognizers by a large margin and with significance.

Wii Remote

In Table 3, we report results for Jackknife variants on Cheema *et al.*'s Wii Remote dataset [7]. This dataset contains 15625 samples of gestures collected from 25 participants who individually provided 25 samples per gesture. This dataset is particularly interesting because as compared to other Wii Remote datasets, the gesture vocabulary is large at 25 classes and the Wii Remote traverses through various orientations. Our results were obtained using a user-dependent protocol

Measure	CF	T=1		T=2	
		μ	(σ)	μ	(σ)
ED	False	0.91	(0.03)	0.92	(0.03)
ED	True	0.91	(0.03)	0.92	(0.03)
IP	False	0.94	(0.02)	0.95	(0.03)
IP	True	0.94	(0.02)	0.95	(0.03)

Effect	$F_{1,63}$	$Pr(>F)$	η_p^2
Measure	412.65	$< .001$	.87 (L)
CF	0.33	.57	.01 —
Templates	86.8	$< .001$	.58 (L)
Measure $\times$ CF	0.0	.96	.00 —
Measure $\times$ Templates	10.84	$< .01$	.15 (L)
CF $\times$ Templates	0.13	.72	.00 —
Measure $\times$ CF $\times$ Templates	0.02	.88	.00 —

Table 1: Writer-independent mean accuracies for Jackknife variants on \$1-GDS [38], as well as ANOVA results with statistically significant effects highlighted and their effect size labeled: (L)arge, (M)edium, or (S)mall.

Recognizer	T=1		T=2	
	μ	(σ)	μ	(σ)
Jackknife	0.94	(0.02)	0.95	(0.03)
\$1	0.88	(0.02)	0.90	(0.01)
\$P	0.84	(0.02)	0.86	(0.02)

Table 2: Writer-independent mean accuracies for several recognizers on \$1-GDS [38]. There is a significant difference between recognizers with a large effect size ($F_{2,45}, p < .001, \eta_p^2 = .92$), and a post hoc analysis shows that all recognizers are significantly different from each other.

similar to the original work of Cheema *et al.* [7]. For each experiment in this protocol, T samples are selected at random from a participant for training and the remaining $25 - T$ samples are selected for testing, and this selection process is performed 500 times per each participant. We did not run a user-independent protocol because device orientation has a significant impact on the accelerometer signal data, and there is a great deal of variance in how the Wii Remote is held by each participant. However, applications that track device orientation can easily adjust the signals in order to support user-independence. To improve accuracy for the inner product (IP) Jackknife variant, the acceleration data was integrated into 3D position points per [15], but we found this was unhelpful for the squared Euclidean distance variant (ED). There was a significant difference between the ED and IP measures, where the IP measure gave higher accuracies. There was also a small positive effect when utilizing the correction factors; though, due to truncation, this cannot be seen in the table.

We next compared the best performing Jackknife variant against alternative domain-specific recognizers (DTW with quantization [18], \$3 [14], and Protractor 3D [15]), whose results are shown in Table 4. DTW with quantized accelerometer data performed slightly better than Jackknife, although accuracies were similar. We note that other than the common step to integrate acceleration samples into position trajectories, Jackknife did not require domain specific knowledge to achieve high accuracy. On the other hand, DTW with quantized data required an analysis of the accelerometer data and a selection of the quantization levels, which may be domain or device specific.

Measure	CF	T=1		T=2	
		μ	(σ)	μ	(σ)
ED	False	0.79	(0.05)	0.86	(0.04)
ED	True	0.80	(0.05)	0.86	(0.04)
IP	False	0.93	(0.03)	0.96	(0.02)
IP	True	0.93	(0.03)	0.96	(0.02)

Effect	$F_{1,168}$	$Pr(>F)$	η_p^2
Measure	1747.23	$< .001$	.91 (L)
CF	3.91	$< .05$	.02 (S)
Templates	314.67	$< .001$	.65 (L)
Measure $\times$ CF	2.23	.14	.01 (S)
Measure $\times$ Templates	44.05	$< .001$	.21 (L)
CF $\times$ Templates	0.17	.68	.00 —
Measure $\times$ CF $\times$ Templates	0.03	.86	.00 —

Table 3: User-dependent mean accuracies for Jackknife variants on Cheema *et al.*'s Wii Remote dataset [7], as well as ANOVA results with statistically significant effects highlighted and their effect size labeled: (L)arge, (M)edium, or (S)mall.

Recognizer	T=1		T=2	
	μ	(σ)	μ	(σ)
DTW (w/ quantization)	0.94	(0.03)	0.97	(0.02)
Jackknife	0.93	(0.03)	0.96	(0.02)
\$3	0.71	(0.06)	0.79	(0.06)
Protractor 3D	0.63	(0.06)	0.73	(0.06)

Table 4: User-dependent mean accuracies for various recognizers on Cheema *et al.*'s Wii Remote dataset [7].The recognizer main effect is significant with a large effect size ($F_{3,168} = 789.72, p < .001, \eta_p^2 = .93(L)$), and post hoc tests show that all recognizers are also significantly different from each other.

Kinect

In Table 5, results are presented for the Ellis *et al.* (Parkour) [9] Kinect dataset, which contains 1280 samples of 16 parkour actions, *e.g.* climbing and vaulting, collected from 16 participants using a Kinect sensor. We used the same user-dependent test protocol as reported in the last section. The local cost function was significant, where the Jackknife IP measure outperformed ED, and IP was able to achieve 99% accuracy with one training sample. The correction factors were are also significant and played a role in substantially driving down the error rates[4]. We also ran a user-independent variant of this test and found that IP with correction factors and T=2 achieved 96% accuracy, whereas ED with correction factors achieved 70%.

We also replicated Ellis *et al.*'s observational latency test [9] that evaluated recognizer accuracy when training and test data

[4]It is important to note that as accuracies reach high levels, seemingly small improvements in accuracy are actually large reductions in error rates.

Measure	CF	T=1		T=2	
		μ	(σ)	μ	(σ)
ED	False	0.88	(0.05)	0.93	(0.03)
ED	True	0.89	(0.05)	0.94	(0.03)
IP	False	0.97	(0.03)	0.99	(0.02)
IP	True	0.99	(0.02)	0.99	(0.01)

Effect	$F_{1,105}$	$Pr(>F)$	η_p^2
Measure	383.89	$< .001$	.79 (L)
CF	16.62	$< .001$	.14 (M)
Templates	80.36	$< .001$	.43 (L)
Measure × CF	1.73	.19	.02 (S)
Measure × Templates	35.86	$< .001$	.25 (L)
CF × Templates	3.38	.07	.03 (S)
Measure × CF × Templates	0.0	.96	.00 —

Table 5: User-dependent mean accuracies for Jackknife variants on the Parkour dataset [9], as well as ANOVA results with statistically significant effects highlighted and their effect size labeled: (L)arge, (M)edium, or (S)mall.

were truncated to varying frame counts in order to minimize the delay between when a user performs an action and when the time that action is recognized. If an action can be recognized before completion, observational latency can be reduced. In Table 6, we see that Jackknife with IP achieves the highest accuracy for all frame count levels.

Recognizer	10	15	20	25	30	40	60
Jackknife IP w/ CF	24	53	78	90	95	98	99
Ellis *et al.* [9]	14	37	65	82	91	95	96
Conditional Random Field	15	25	47	67	81	91	94
Bag of Words	11	21	44	68	83	92	94

Table 6: Recognition percentage accuracies for Jackknife and recognizers evaluated by Ellis *et al.* [9] (bottom three) for varying length video sequences. Both training and testing data are truncated to the specified number of frames.

Acoustic Gestures

Hand based gesture interactions with computers via Doppler shifted sound waves is presently gaining attention [11, 27]. Unlike other interface devices evaluated to this point, sound waves are especially subject to noise as extracting and detecting frequency changes over a large spectrum with low cost hardware still lacks robustness. Further, complex over-the-air hand gestures are prone to large variations in shape and speed, the later of which manifests itself uniquely in frequency distributions (unlike in 2D or 3D Euclidean space where speed alone does not change the observed trajectory). These issues make accurate gesture recognition difficult. To test whether Jackknife is suitable for this input modality, we collected a new dataset comprising eighteen dynamic hand gestures collected from 22 participants with a 5 speaker, 1 microphone setup based on [22] (see our project website for more details). The raw data is represented as a 33 frequency bin distribution per speaker, which results in an $m = 165$ component vector per frame. Jackknife results are shown in Table 7, which were obtained using the user-dependent protocol [7] described previously. In early testing, we learned that z-score normalization on the spectrum data was harmful; we believe this is because for some gestures, there is no motion through certain frequency bins, and so z-score normalizing those components only served to scale up noise. Therefore, ED- is also reported, which is the squared Euclidean distance measure on raw data without z-score normalization. While ED- is not formally part of Jackknife (as this result required an additional investigation and some domain knowledge), we feel that the reader should be aware of this result. Additionally, since the bounding box correction factor does not have meaning in this context, we only utilize the absolute distances traveled correction factor.

Measure	CF	T=1		T=2		T=4	
		μ	(σ)	μ	(σ)	μ	(σ)
ED	False	0.71	(0.08)	0.81	(0.08)	0.87	(0.07)
ED	True	0.75	(0.07)	0.83	(0.06)	0.89	(0.05)
ED-	False	0.83	(0.07)	0.90	(0.05)	0.94	(0.04)
ED-	True	0.84	(0.06)	0.91	(0.05)	0.94	(0.04)
IP	False	0.72	(0.07)	0.81	(0.06)	0.88	(0.06)
IP	True	0.75	(0.06)	0.84	(0.05)	0.90	(0.05)

Effect	F Value	$Pr(>F)$	η_p^2
Measure	$F_{2,483} = 491.46$	$< .001$	.67 (L)
CF	$F_{1,483} = 72.36$	$< .001$	.13 (M)
Templates	$F_{3,483} = 551.31$	$< .001$	.77 (L)
Measure × CF	$F_{2,483} = 10.02$	$< .001$	.04 (S)
Measure × Templates	$F_{6,483} = 11.41$	$< .001$	.12 (M)
CF × Templates	$F_{3,483} = 1.76$	.15	.01 (S)

Table 7: User-dependent accuracy results for Jackknife variants on the acoustic gesture dataset, as well as ANOVA results with statistically significant effects highlighted and their effect size labeled: (L)arge, (M)edium, or (S)mall. ED- indicates that sequences were not z-score normalized.

As is shown, the local distance measure and correction factor effects are significant. In a post hoc analysis, we found that ED- was significantly different from ED and IP, but the latter measures where not different from each other. Good accuracy ($\geq 90\%$) can be achieved with ED- with two templates, otherwise four templates are required with IP.

Performance

Over 10,000 iterations, we trained a C++ based Jackknife recognizer with our Kinect user study data (see next section), which is 63 components per point. The recognizer was trained with T=10 templates per gesture class, resulting in 140 templates (since there are 14 gesture classes). The recognizer was then used to evaluate one additional sample per gesture class and the evaluation time in *micro*seconds per sample was recorded. On a MacBook Pro, 2.2 GHz Intel Core i7 with 8GB DDR3 memory, the average time to execute a recognition test on a raw sample was 395 µs (std=90.2); or equivalently, the evaluation process took approximately 2.82 µs (0.64) per template, amortized.

EVALUATION OF CONTINUOUS DATA

To evaluate the effectiveness of our approach in rejecting non-gesture sequences from a continuous data stream, we collected

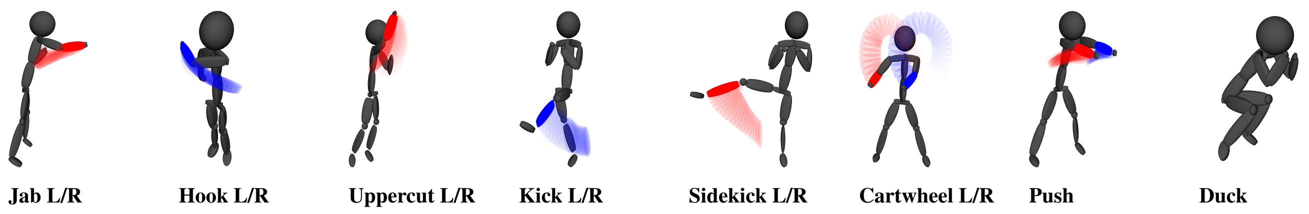

Table 8: List of 14 Kinect gestures used in our contiguous data study. Note that L/R indicates there is a left and right side version of the gesture.

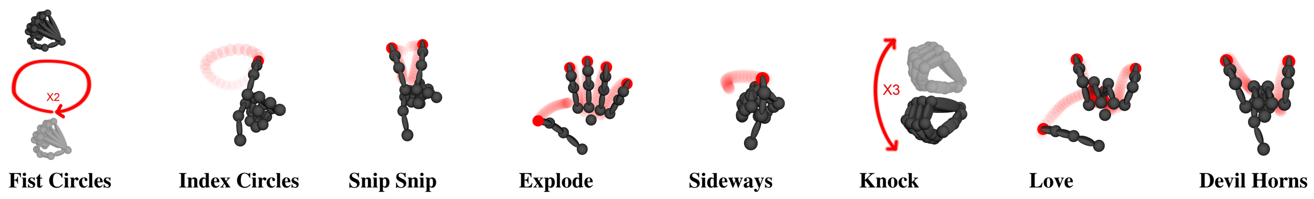

Table 9: List of 8 Leap Motion gestures used in our contiguous data study. Each gestures starts with one's hand in a first and ends in the same position.

λ	F_1 Score	Precision	Recall	FPR	Pruned
	$\mu(\sigma)$	$\mu(\sigma)$	$\mu(\sigma)$	$\mu(\sigma)$	$\mu(\sigma)$
1.5	95.7 (5)	95.4 (5)	96.1 (6)	0.3 (0)	89.2 (6)
1.75	96.0 (6)	96.0 (6)	96.1 (7)	0.3 (0)	90.1 (5)
2.0	96.9 (5)	97.3 (5)	96.5 (6)	0.2 (0)	90.9 (5)
2.25	96.4 (6)	97.5 (4)	95.5 (7)	0.2 (0)	91.7 (5)
2.5	95.9 (6)	97.5 (4)	94.5 (8)	0.2 (0)	92.6 (4)
-2.75	95.1 (5)	97.5 (3)	93.0 (8)	0.2 (0)	93.3 (4)
1.98	96.4 (5)	96.8 (5)	96.1 (6)	0.2 (0)	90.8 (5)

Table 10: T=2 Percentage accuracy results for various rejection thresholds on the Kinect gesture dataset shown in Table 8. The last entry is the threshold that was automatically selected.

λ	F_1 Score	Precision	Recall	FPR	Pruned
	$\mu(\sigma)$	$\mu(\sigma)$	$\mu(\sigma)$	$\mu(\sigma)$	$\mu(\sigma)$
1.5	93.8 (5)	93.3 (7)	95.0 (7)	0.8 (0)	54.8 (7)
1.75	94.4 (5)	95.0 (5)	94.4 (8)	0.6 (0)	58.1 (7)
2.0	94.3 (5)	95.8 (5)	93.3 (8)	0.5 (0)	62.5 (7)
2.25	94.0 (7)	97.7 (3)	91.5 (11)	0.3 (0)	66.2 (6)
2.5	92.8 (8)	98.1 (3)	89.4 (13)	0.2 (0)	70.2 (6)
-2.75	90.9 (10)	98.3 (2)	86.3 (15)	0.2 (0)	73.7 (5)
1.95	93.8 (6)	95.7 (5)	92.5 (9)	0.5 (0)	61.8 (8)

Table 11: T=2 Percentage accuracy results for various rejection thresholds on the Leap Motion gesture dataset shown in Table 9. The last entry is the threshold that was automatically selected.

test data from a pool of 40 students (30 male and 10 female) at the University of Central Florida, ranging in age from 18 to 28. The participants were divided into two groups where the first group worked with a Kinect and the second group worked with a Leap Motion. We collected segmented training data of the gestures shown in Tables 8 and 9 as well as a continuous, uninterrupted session of the sample gestures performed in random order with repetitions, which is discussed in detail below. Each participant took around 15 to 20 minutes to complete all tasks, including a pre-questionnaire to collect demographic information, and all were compensated $10 for their time.

The experimental setup comprised a 50 inch Sony Bravia HDTV and a Microsoft Kinect 2.0 or Leap Motion. The Kinect sensor was mounted above the HDTV using a mounting device and was kept stationary throughout all sessions. The Leap Motion sensor was mounted on a desktop in front of the television and was kept in place through a single session using tape, except in one case where the participant was left handed and the sensor's orientation was changed to accommodate for their comfort. Further, a box was placed between the participant and the device so one could rest their arm and avoid fatigue, which also helped to control the distance and orientation of a participant's hand during the study.

Data Collection

We developed a data collection utility with two modes using Unity 3D v5.4. The first mode allowed us to collect clean, segmented samples of each gesture for training. A participant was asked to perform a specific gesture that was initiated by a count down. The sequence was then resampled and replayed to ensure correctness. If an error occurred or the participant was unhappy with the replay, the sample was discard and recollected. This process was repeated twice so two training samples per gesture were collected. Note that all gestures were demonstrated to a participant before any data was collected. The second mode enabled us to collect a continuous stream of gesture articulations. Our application prompted the participant to perform a specific gesture selected at random, and once we confirmed the correct action was performed, we manually advanced the application to the next gesture. Otherwise, if an error occurred, *e.g.*, a malformed gesture or loss of tracking, the action was retried. A proper example of each gesture was collected three times so that there were a total of 14×3 valid Kinect gestures and 8×3 valid Leap Motion gestures included in the *unsegmented* stream.

We noticed the Kinect lost track of some participants more frequently than others, which may be related to the texture and colors of the participants clothing. In such cases the skeleton appeared jittery on the screen and gesture were repeated as necessary. The Leap Motion device also frequently lost tracking and detected incorrect hand poses, such as reporting that both the index and middle fingers were extended when, in fact, only the index was so. These problems were exacerbated with 5 participants who had smaller hands. Also, we actually collected data for 9 Leap Motion gestures, but found one gesture (not shown in Table 9) was difficult for participants to perform. This gesture was removed from our analysis.

Results

Using data collected from each participant, we trained Jackknife and replayed the participant's session. All classification results including true and false positives (tp and fp) as well as true and false negatives (fn and fn) were averaged into an overall result. Additional parameters were tuned as follows: the recognizer was run once every 10 frames using the last four seconds of buffered frame data[5]; once an action was detected, the buffer was cleared and the recognizer was suspended for 2 seconds, which we believe is sufficient time to prepare for the next gesture; and a gesture was considered as executed if Jackknife returned the same result twice in a row. Based on our experiences, these are fairly reasonable criteria, which can be tuned to match a practitioner's specific requirements.

We reran the above procedure several times for different levels of λ used to set the rejection threshold (see Equation 12). Table 10 shows results for our Kinect continuous data test, and Table 11 shows results for the Leap Motion test. The rejection threshold is important in balancing precision $(tp/(tp+fp))$ and recall $(tp/(tp+fn))$, and the F_1 score is the harmonic mean of these measures. Our goal is to therefore maximum F_1. In the Kinect test, we are able to achieve 96.9% at $\lambda = 2.0$, which is close to the automatically selected threshold $\lambda = 1.98$. Similarly, the maximum F_1 score of 94.4% for the Leap Motion test occurred at $\lambda = 1.75$, which again is near the automatically selected $\lambda = 1.95$. Recall that the automatic threshold is selected so that the false and true negatives generated from synthetic data is minimized, which appear to be appropriate for idle movement in between actions. However, λ can also be increased to allow for a larger variety of non-gesture actions to be performed at the expense increased false negatives. According to our results, the affect on the F_1 score is not too large, where the improvement in precision may well be worth the loss in recall for many applications. These results suggest that Jackknife is useful for working with continuous data.

DISCUSSION

Jackknife is designed to be a "go-to" recognizer for gesture interface development that can span multiple modalities and still achieve competitive accuracy. By combining DTW with concepts developed for 2D gesture recognition, for rapid prototyping and gesture customization, we believe our recognizer accomplishes its objective. There is no simple criterion by which one can judge a recognizer and determine that is it excellent or terrible. At best we can compare with other recognizers in a specific domain for a specific dataset when such is available and speak in relative terms. In this way, we see that Jackknife is very competitive. Examples include the \$1-GDS pen and touch dataset where Jackknife with IP outperformed \$1 and \$P; Cheema *et al.*'s [7] Wii Remote datasets, where Jackknife was on par with the domain-specific quantized DTW; and Ellis *et al.*'s [9] Parkour dataset, where our recognizer achieved 99% accuracy with one training sample, and can be used to reduce observational latency. Even with atypical gesture data such as sound frequencies, Jackknife is able to reach greater than 90% accuracy with only two training samples per gesture. From another perspective, based on accuracy results reported for 36 3D gesture recognizers across different domains [16] (Table 1), one might expect a competitive recognizer to fall between 88—98% ($93\% \pm 5.29\%$) accuracy. In all tests performed, Jackknife fell in this range.

However, a gesture recognizer can only be a go-to option if it can also support continuous data, which requires robust rejection. By using GPSR [32] to create synthetic positive samples that are combined with synthetic negative samples, we are able to learn a per template rejection threshold with only one training sample per gesture class. We were able to show through an evaluation of continuous data that this automatic process is able to derive a threshold near the optimum.

One Recognizer To Rule Them All?

No, while Jackknife has been demonstrated to be a versatile tool, we do not claim that our recognizer is appropriate for every situation or even that it is the best solution possible for any one situation, and some limitations are worth noting. First, Jackknife presently does not handle static poses. Motion into and out of a static pose can be detected, but a single frame pose does not constitute a time series appropriate for DTW treatment, a limitation of Jackknife we intend to address in future work. The techniques presented also do not generalize to arbitrary variability; for instance, a 3D hand gesture of a circle drawn counterclockwise does not automatically pair with a clockwise gesture without providing appropriate training samples. A last issue that requires attention is that the resampling rate n for GPSR has to be manually tuned. Similar to the optimal-n equation for synthetic 2D gestures, work needs to be done to find an equivalent for more complex data.

CONCLUSION

We have presented Jackknife, a general gesture recognizer suitable for rapid prototyping and gesture customization that works well with little training data as well as continuous data. Our recognizer uses DTW with an inner product local cost function on normalized gesture path direction vectors, which was shown to outperform the squared Euclidean distance alternative in most tests, although Jackknife can be equipped with either measure. We also introduced two new correction factors for DTW that help disambiguate certain gesture classes and inflate negative, non-gesture sample scores. Finally, we proposed a new method of automatically selecting a rejection criteria for continuous data that, according to our evaluation of online data, can select a threshold close to the optimum. Overall, we have demonstrated that Jackknife shows promise of being a capable and robust go-to gesture recognizer.

[5]We used four seconds because some gestures were performed slowly by some participants; though we could have used a shorter duration in most cases. Since the gesture paths are resampled to $n = 16$, idle frames do not significantly contribute to the shape of the action.

ACKNOWLEDGMENTS
This work is supported in part by NSF Award IIS-1638060, Lockheed Martin, Office of Naval Research Award ONR-BAA15001, Army RDECOM Award W911QX13C0052, and Coda Enterprises, LLC. We also thank the ISUE lab members at UCF for their support as well as the anonymous reviewers for their helpful feedback.

REFERENCES
1. Lisa Anthony and Jacob O. Wobbrock. 2010. A Lightweight Multistroke Recognizer for User Interface Prototypes. In *Proceedings of Graphics Interface 2010 (GI '10)*. Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 245–252.
2. Lisa Anthony and Jacob O. Wobbrock. 2012. $N-protractor: A Fast and Accurate Multistroke Recognizer. In *Proceedings of Graphics Interface 2012 (GI '12)*. Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 117–120.
3. Anthony Bagnall, Aaron Bostrom, James Large, and Jason Lines. 2016. The Great Time Series Classification Bake Off: An Experimental Evaluation of Recently Proposed Algorithms. Extended Version. *arXiv preprint arXiv:1602.01711* (2016).
4. Gustavo E.A.P.A. Batista, Xiaoyue Wang, and Eamonn J. Keogh. 2011. *A Complexity-Invariant Distance Measure for Time Series*. Chapter 60, 699–710.
5. Saša Bodiroža, Guillaume Doisy, and Verena Vanessa Hafner. 2013. Position-invariant, Real-time Gesture Recognition Based on Dynamic Time Warping. In *Proceedings of the 8th ACM/IEEE International Conference on Human-robot Interaction (HRI '13)*. IEEE Press, Piscataway, NJ, USA, 87–88.
6. Sait Celebi, Ali Selman Aydin, Talha Tarik Temiz, and Tarik Arici. 2013. Gesture Recognition using Skeleton Data with Weighted Dynamic Time Warping.. In *VISAPP (1)*. 620–625.
7. Salman Cheema, Michael Hoffman, and Joseph J. LaViola Jr. 2013. 3D Gesture classification with linear acceleration and angular velocity sensing devices for video games. *Entertainment Computing* 4, 1 (2013), 11 – 24.
8. Heng Du and TszHang To. 2011. Hand gesture recognition using Kinect. *Techical Report, Boston University* (2011).
9. Chris Ellis, Syed Zain Masood, Marshall F. Tappen, Joseph J. Laviola, Jr., and Rahul Sukthankar. 2013. Exploring the Trade-off Between Accuracy and Observational Latency in Action Recognition. *International Journal of Computer Vision* 101, 3 (Feb. 2013), 420–436.
10. Rafael Giusti and Gustavo E. A. P. A. Batista. 2013. An Empirical Comparison of Dissimilarity Measures for Time Series Classification. In *Proceedings of the 2013 Brazilian Conference on Intelligent Systems (BRACIS '13)*. IEEE Computer Society, Washington, DC, USA, 82–88.
11. Sidhant Gupta, Daniel Morris, Shwetak Patel, and Desney Tan. 2012. Soundwave: using the doppler effect to sense gestures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1911–1914.
12. Noor A. Ibraheem and Rafiqul Z. Khan. 2012. Survey on Various Gesture Recognition Technologies and Techniques. *International Journal of Computer Applications* 50, 7 (2012).
13. Eamonn Keogh and Chotirat Ann Ratanamahatana. 2005. Exact indexing of dynamic time warping. *Knowledge and information systems* 7, 3 (2005), 358–386.
14. Sven Kratz and Michael Rohs. 2010. The $3 recognizer: simple 3D gesture recognition on mobile devices. In *Proceedings of the 15th international conference on Intelligent user interfaces*. ACM, 419–420.
15. Sven Kratz and Michael Rohs. 2011. Protractor3D: a closed-form solution to rotation-invariant 3D gestures. In *Proceedings of the 16th international conference on Intelligent User Interfaces*. ACM, 371–374.
16. Joseph J LaViola. 2013. 3d gestural interaction: The state of the field. *International Scholarly Research Notices* 2013 (2013).
17. Yang Li. 2010. Protractor: A Fast and Accurate Gesture Recognizer. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. ACM, New York, NY, USA, 2169–2172.
18. Jiayang Liu, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. 2009. uWave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing* 5, 6 (2009), 657 – 675. PerCom 2009.
19. Xiao-Hui Liu and Chin-Seng Chua. 2010. Rejection of non-meaningful activities for HMM-based activity recognition system. *Image and Vision Computing* 28, 6 (2010), 865–871.
20. Robert Macrae and Simon Dixon. 2010. Accurate real-time windowed time warping. In *in ISMIR, 2010*. 423–428.
21. S. Mitra and T. Acharya. 2007. Gesture Recognition: A Survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 37, 3 (May 2007), 311–324.
22. Corey Pittman, Pamela Wisniewski, Conner Brooks, and Joseph J. LaViola Jr. 2016. Multiwave: Doppler Effect Based Gesture Recognition in Multiple Dimensions. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. ACM, 1729–1736.

23. Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. 2012. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 262–270.

24. Chotirat Ann Ratanamahatana and Eamonn Keogh. 2004. Everything you know about dynamic time warping is wrong. In *Third Workshop on Mining Temporal and Sequential Data*.

25. Siddharth S. Rautaray and Anupam Agrawal. 2015. Vision based hand gesture recognition for human computer interaction: a survey. *Artificial Intelligence Review* 43, 1 (2015), 1–54.

26. Douglas A Reynolds, Thomas F Quatieri, and Robert B Dunn. 2000. Speaker verification using adapted Gaussian mixture models. *Digital signal processing* 10, 1 (2000), 19–41.

27. Wenjie Ruan, Quan Z Sheng, Lei Yang, Tao Gu, Peipei Xu, and Longfei Shangguan. 2016. AudioGest: enabling fine-grained hand gesture detection by decoding echo signal. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 474–485.

28. Dean Rubine. 1991. Specifying Gestures by Example. *SIGGRAPH Computer Graphics* 25, 4 (July 1991), 329–337.

29. Hiroaki Sakoe and Seibi Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing* 26, 1 (1978), 43–49.

30. Arpita R. Sarkar, G. Sanyal, and S. Majumder. 2013. Hand Gesture Recognition Systems: A Survey. *International Journal of Computer Applications* 71, 15 (2013).

31. J. Suarez and R. R. Murphy. 2012. Hand gesture recognition with depth images: A review. In *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*. 411–417.

32. Eugene M. Taranta, II, Mehran Maghoumi, Corey R. Pittman, and Joseph J. LaViola, Jr. 2016. A Rapid Prototyping Approach to Synthetic Data Generation for Improved 2D Gesture Recognition. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16)*. ACM, New York, NY, USA, 873–885.

33. Eugene M. Taranta II, Andrés N. Vargas, and Joseph J. LaViola Jr. 2016. Streamlined and accurate gesture recognition with Penny Pincher. *Computers & Graphics* 55 (2016), 130 – 142.

34. Radu-Daniel Vatavu. 2011. The Effect of Sampling Rate on the Performance of Template-based Gesture Recognizers. In *Proceedings of the 13th International Conference on Multimodal Interfaces (ICMI '11)*. ACM, New York, NY, USA, 271–278.

35. Radu-Daniel Vatavu, Lisa Anthony, and Jacob O. Wobbrock. 2012. Gestures As Point Clouds: A $P Recognizer for User Interface Prototypes. In *Proceedings of the 14th ACM International Conference on Multimodal Interaction (ICMI '12)*. ACM, New York, NY, USA, 273–280.

36. Sharad Vikram, Lei Li, and Stuart Russell. 2013. Writing and Sketching in the Air, Recognizing and Controlling on the Fly. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems (CHI EA '13)*. ACM, New York, NY, USA, 1179–1184.

37. Jacob O. Wobbrock, Leah Findlater, Darren Gergle, and James J. Higgins. 2011. The Aligned Rank Transform for Nonparametric Factorial Analyses Using Only Anova Procedures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 143–146.

38. Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. 2007. Gestures Without Libraries, Toolkits or Training: A $1 Recognizer for User Interface Prototypes. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology (UIST '07)*. ACM, New York, NY, USA, 159–168.

39. J. Wu, J. Konrad, and P. Ishwar. 2013. Dynamic time warping for gesture-based user identification and authentication with Kinect. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2371–2375.

40. Yaodong Zhang and James R Glass. 2011. An inner-product lower-bound estimate for dynamic time warping. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 5660–5663.